Testbed-12 WFS 2.0 CITE and Reference Implementation Installation User Guide

Table of Contents

1. WFS 2.0 capabilities document and conformance classes 4
2. Installation Guide for Deegree 3 as WFS 2.0 RI 5
   2.1. Java SE 5
   2.2. Apache Tomcat 5
   2.3. PostgreSQL 5
   2.4. PostGIS 6
   2.5. deegree 6
   2.6. deegree workspace 6
   2.7. Short introduction into usage of deegree 7
3. Best practices to implement WFS 2.0 8
   3.1. Coordination with test developer for proper interpretation of standard 8
   3.2. Do not underestimate low-level conformance classes 8
   3.3. Process of implementing a reference implementation also improves quality of corresponding CITE test suite 9
   3.4. Joins 9
   3.5. Complex Properties 9
   3.6. Time Zones 10
   3.7. deegree configuration options must be understood 11
4. Installation and integration with CITE validation tools 12
   4.1. Installation and setup of TEAM Engine and WFS 2.0 test suite 12
      4.1.1. Requirements 12
      4.1.2. Installation of TEAM Engine 12
      4.1.3. Installation of WFS 2.0 test suite 13
   4.2. Test WFS 2.0 reference implementation with WFS 2.0 test suite 14
      4.2.1. Execute a test of the reference implementation 14
      4.2.2. Examine the results of the test run 15
5. Suggestions for change requests deriving from realization of reference implementation 16
Appendix A: Revision history 17

Publication Date: 2017-06-16

Approval Date: 2017-03-23

Posted Date: 2016-01-01

Reference number of this document: OGC 16-025r2

Reference URL for this document: http://www.opengis.net/doc/bp/t12-A089

Category: User Guide

Editor: D. Stenger, L. Goltz, J. Fitzke

Title: Testbed-12 WFS 2.0 CITE and Reference Implementation Installation User Guide

---

NOTE

This document is a user guide created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard. It is not an official position of the OGC membership. There may be additional valid approaches beyond what is described in this user guide.

Abstract

The WFS 2.0 CITE and Reference Implementation (RI) Installation User Guide documents the results of the WFS 2.0 Reference Implementation development in the OGC Testbed 12 Compliance (CMP) thread.

The following content is covered by this engineering report:

- WFS 2.0 capabilities document and conformance classes
- Installation Guide for Deegree 3 as WFS 2.0 RI
- Best practices to implement WFS 2.0
- Installation and integration with CITE validation tools

Importance to OGC WFS Working Groups and CITE SC

Within the OGC Testbed 12 Compliance (CMP) thread, the WFS 2.0 test suite and a WFS 2.0 reference implementation were enhanced for a complete coverage of the WFS 2.0 requirements. It is expected that this activity will yield insight into specific compliance capabilities and help discover issues in implementations of software for WFS 2.0. The following test suite requirements were advanced:

- Transactional WFS – Ability to create and update capabilities. The current test already supports transactions, which is restricted to features.
- Locking WFS – Ability to allow multiple users to update simultaneously the data provided bv the WF
- Response Paging – Paging is supported in CSW and Content Discovery and Retrieval (CDR) applications. The test was advanced to be consistent with the WFS 2.0 standard and to be compatible and consistent with CDR.
- Standard Joins – ComparisonOperators can be used for joins. The test and reference implementation include the case of multi-source integration.
- Spatial Joins – SpatialOperators can be used for joins. The test and reference implementation include the case of multi-source integration.
- Temporal Joins – TemporalOperators can be used for joins. The test and reference implementation include multi-source integration.
- Feature Versions – The test and reference implementation test Object Based Production, which is producing feature objects and versioning them.
- Manage Stored Queries - Stored Queries can be created and dropped. Creation of illegal Stored Queries is prevented.

POINTS OF CONTACT

Name	Organization
D. Stenger, L. Goltz, J. Fitzke (ed.)	lat/lon GmbH
Luis Bermudez	OGC

Chapter 1. WFS 2.0 capabilities document and conformance classes

A product implementing WFS 2.0 needs to advertise the conformance classes in the capabilities document. If the capability (e.g. Spatial Joins) is not advertises by the server, clients (e.g. the OGC TEAM Engine validation tool) will not know that the server has implemented that conformance class.

The WFS 2.0 reference implementation is compliant to the following conformance classes. All conformance classes are described in "OpenGIS Web Feature Service 2.0 Interface Standard (also ISO 19142)" (document 09-025r1).

- **Transactional WFS** (Chapter "A.1.3 Transactional WFS", page 101)
- **Locking WFS** (Chapter "A.1.4 Locking WFS", page 102)
- **Response Paging** (Chapter "A.1.10 Response Paging", page 103)
- **Standard Joins** (Chapter "A.1.11 Standard Joins", page 103)
- **Spatial Joins** (Chapter "A.1.12 Spatial Joins", page 103)
- **Temporal Joins** (Chapter "A.1.13 Temporal Joins", page 104)
- **Feature Versions** (Chapter "A.1.14 Feature Versions", page 104)
- **Manage Stored Queries** (Chapter "A.1.15 Manage Stored Queries", page 104)

All features and interfaces work as described in the "OpenGIS Web Feature Service 2.0 Interface Standard (also ISO 19142)".

Chapter 2. Installation Guide for Deegree 3 as WFS 2.0 RI

The WFS 2.0 RI is based on the deegree software. The deegree software provides workspaces that are preconfigured for ease of use. A deegree workspace was made available configured with test data from Iceland supporting all features required for WFS 2.0 (See bellow).

A general installation guide for deegree can be found in the handbook. Restrictions regarding the operating system, third party software and other dependencies are also documented there.

It is recommended to use a Debian based operating system Ubuntu. The following chapters regarding the installation assume that a Debian based operating system is used.

The following software components have to be installed and configured to setup and run the reference implementation.

- Java SE
- Apache Tomcat
- PostgreSQL
- PostGIS
- deegree
- deegree workspace

2.1. Java SE

Recommended Version: Oracle Java 8 (JDK)

Download and install.

2.2. Apache Tomcat

Recommended Version: Apache 7

Install Apache Tomcat via the packaging managing system of the operation system:

```
apt-get install tomcat7
```

2.3. PostgreSQL

Recommended Version: PostgreSQL 9.4

Install PostgreSQL via the packaging managing system of the operation system:

```
apt-get install postgresql-9.4
```

2.4. PostGIS

Recommended Version: PostGIS 2.1 or 2.2

Install PostGIS via the packaging managing system of the operation system:

```
apt-get install postgis
```

2.5. deegree

deegree artifact is provided via a Nexus repository

Download the war file, rename it to deegree.war and put it into the webapps folder of the Apache Tomcat:

```
wget -O deegree.war ${deegree.download.url}
mv deegree.war ${tomcat}/webapps
```

After starting the tomcat deegree will be available at:

```
http://HOST:PORT/deegree.
```

2.6. deegree workspace

deegree workspace artifact is provided via a Nexus repository

Download the deegree workspace and unzip it into the DEEGREE_WORKSPACE_ROOT folder (should be /home/USER/.deegree):

```
wget -O workspace-iceland.zip ${workspace.download.url}
unzip workspace-iceland.zip -d ${DEEGREE_WORKSPACE_ROOT}
```

Import the test data as described below and activate the workspace via the deegree console at:

```
http://HOST:PORT/deegree
```

Import of the testdata:

1. Switch to the directory *sql* in the workspace folder (should be /home/USER/.deegree/workspace-iceland)

2. Execute the script install-db.sh

```
./install-db.sh
```

TIP	If you are the user 'postgres' and the PostgreSQL database is installed with default settings, the script will work without any adjustments.

2.7. Short introduction into usage of deegree

Open deegree in your web browser by typing:

```
http://HOST:PORT/deegree
```

You can see the deegree console which can be used to configure deegree and to find out all relevant information about the configured services (e.g. endpoint URLs).

For the RI the most important basics of deegree are:

- How to initialize a workspace.
- How to determine endpoint URLs.
- How the configuration concept of deegree works in general.

How to initialize a workspace:

Go to menu item 'workspaces'. In the list 'Available workspaces' all existing workspaces are shown. Choose a workspace and click on 'Start'. In case you want to activate the WFS 2.0 RI workspace pick 'workspace-iceland'. deegree is now initializing and starting the selected workspace which can take a couple of seconds. Finally, on the top of the deegree console the active workspace is shown (e.g. 'Active workspace: workspace-iceland').

How to determine endpoint URLs:

Go to menu item 'services'. There you see a list of services (in case the WFS 2.0 RI workspace is active only one), each with the button 'Capabilities'. Click on the button and the Capabilities document of the service opens in a new tab. This URL is important for the CITE test execution later.

How the configuration concept of deegree works in general:

All configuration files of deegree are written in XML. They can be accessed via following menu items (list just contains configuration files which are important for a WFS 2.0): services, feature and databases.

Chapter 3. Best practices to implement WFS 2.0

The process of implementing the WFS 2.0 reference implementation with deegree led to several outcomes regarding the process itself.

This chapter describes the lessons learned by formulating best practices.

3.1. Coordination with test developer for proper interpretation of standard

Coordination and communication with other Testbed 12 participants, in particular the test developer is important. Implementers can be challenged with the task of interpreting the specification, as it is sometimes not precise enough for software developers, especially for more advanced features. In this case various communication channels (e.g. conversation at the issue tracker, mailing lists, text chats, voice chats) can be used to coordinate and to lead discussions with users, other developers and maintainers of the corresponding CITE test suite.

The process of communication leads to a better understanding of a feature for all sides. It also helps implementers of the reference implementation and of the executable test suite to improve the quality of their products. Users contribute the view how a feature is used in cases of application which should absolutely be incorporated by developers. The outcome of the communication process is a better understanding of the specification and a feature for all sides.

A GitHub issue tracker was used to keep track of the issues and the discussions around those issues. The weekly meetings served as a venue to further discuss and reach a solution faster.

A good example that required further clarification for the WFS 2.0 reference implementation was test 'updateSupersededVersion' that is part of the Feature Versions conformance class. See GitHub discussion for more details.

3.2. Do not underestimate low-level conformance classes

If low-level conformance classes (e.g. Simple or Basic) are not implemented very accurate and specific, more specialized conformance classes (e.g. Joins) cannot be passed.

A good example regarding the WFS 2.0 are the filter conformance classes which are an important dependency for the joins conformance classes. So, if the reference implementation does not pass the filter conformance classes it will never pass the joins conformance classes.

3.3. Process of implementing a reference implementation also improves quality of corresponding CITE test suite

During the implementation process of the WFS 2.0 reference implementation several bugs and incompleteness were detected in the ETS. Bug issues were created for that which help the participant developing the ETS WFS 2.0.

By this, the implementation process of the reference implementation improves the quality of an executable test suite as well. So, there is helpful feedback in both directions. The executable test suite improves the quality of the reference implementation and vice versa.

3.4. Joins

When implementing joins two aspects have to be kept in mind:

- All operators which are required by filter conformance classes must be supported by joins as well. This requirement derives from the fact that operators listed in the capabilities document cannot be linked directly to filter and/or join functionality. Thus, all listed operators have to be supported by filters and joins.
- Self joins are a requirement and have to be considered during implementation. A self join is a join operation with two times the same FeatureType.

3.5. Complex Properties

It is necessary to make complex properties available when developing a reference implementation. A time period property has to be introduced to pass the temporal filter and Temporal Joins conformance classes. This property is mandatory for the During operator, which is required by the minimal temporal filter conformance class.

The deegree reference implementation uses the type "gml:TimePeriodPropertyType" to meet this requirement. It can be enabled by adding a complex element to the SQLFeatureStore configuration in deegree.

Example:

```
    <Complex path="app:period">
      <Complex path="gml:TimePeriod">
        <Primitive path="@gml:id" mapping="osm_timestamp_modified_id" />
        <Primitive path="gml:beginPosition" mapping="osm_timestamp" type="dateTime">
          <CustomConverter
class="de.latlon.ogctestbed12.sql.converter.UtcTimeConverter" />
        </Primitive>
        <Primitive path="gml:endPosition" mapping="osm_timestamp_modified"
type="dateTime">
          <CustomConverter
class="de.latlon.ogctestbed12.sql.converter.UtcTimeConverter" />
        </Primitive>
      </Complex>
    </Complex>
```

Another complex property is, of course, introduced by any geometry element (e.g. gml:PointPropertyType or gml:SurfacePropertyType).

3.6. Time Zones

To ensure the possible passing of all temporal related tests, the WFS 2.0 implementation must support time zones in temporal data.

There are two dimensions of this requirement:

- The service implementation must consider time zones when processing a request. This means that time zones must be read from data source and be used for all internal algorithms.
- The GML output of a GetFeature response must include time zones.

Prior to the Testbed 12 project deegree just supported the first dimension of the requirement. The second demand was implemented during this project. The main challenge here was how to deal with time data which do not provide a time zone in data source. For processing deegree uses the default time zone of the server. This behaviour does not satisfy the GML output issue as a user does not want to retrieve time data with the time zone of the server.

Thus, it was implemented that the display of time zones can be configured in the SQLFeatureStore of the deegree workspace. By using the UtcTimeConverter, which was developed for Testbed 12, the display of output of time zones can be controlled.

Example:

```
    <Primitive path="app:osm_timestamp" mapping="osm_timestamp">
      <CustomConverter class="de.latlon.ogctestbed12.sql.converter.UtcTimeConverter"
/>
    </Primitive>
```

This solution satisfies both needs of this requirement.

3.7. deegree configuration options must be understood

In the following a few important configuration options of deegree are explained with regard to the implementation of a WFS 2.0 reference implementation. All of them are described in the handbook.

Table 1. deegree configuration options

configuration option	Value	Description	WFS RI
EnableTransactions	true	Enables WFS-T	Required for Transactional and Locking WFS CC
EnableResponseBufferi ng	true	Enables buffering of the full response before writing it to the response stream	Otherwise invalid exception messages are returned
EnableResponsePaging	true	Enables Response Paging	Required for Response Paging CC
GMLFormat/ GetFeatureResponse/ DisableStreaming	true	Enables collecting of matching features	Otherwise the number of features in a feature collection is unknown as well as the bbox.

Chapter 4. Installation and integration with CITE validation tools

The WFS 2.0 reference implementation can be tested with the WFS 2.0 test suite executable by the TEAM Engine.

The OGC Compliance Program (CITE) advertises those test suites in the OGC validation website.

4.1. Installation and setup of TEAM Engine and WFS 2.0 test suite

An installation guide for the TEAM Engine and how to integrate test suites can be found at the TEAM Engine guides.

4.1.1. Requirements

The TEAM Engine needs a similar software setup like deegree. Java SE and Apache Tomcat have to be installed. For details see chapter "Documentation of WFS RI installation".

In addition, the Maven tool has to be installed.

All requirement for TEAM Engine are existent, when those three software components are installed.

4.1.2. Installation of TEAM Engine

Do a git clone of the TEAM Engine Source Code:

```
git clone https://github.com/opengeospatial/teamengine.git
```

Checkout the desired version of TEAM Engine. We recommend the newest release (currently 4.8):

```
git checkout ${project.version}
```

Build the source code with Maven:

```
mvn clean install -Dmaven.site.skip=true
```

Go to 'target' folder of 'teamengine-web' module:

```
cd teamengine-web/target/
```

Unzip 'teamengine-common-libs.zip' to 'lib' folder of tomcat:

```
unzip -o teamengine-common-libs.zip -d ${tomcat}/lib
```

Copy 'teamengine.war' to 'webapps' folder of tomcat:

```
cp teamengine.war ${tomcat}/webapps
```

Go to 'target' folder of 'teamengine-console' module:

```
cd teamengine-console/target/
```

Unzip 'te_base' folder to home directory:

```
unzip -o teamengine-console-*-base.zip -d ~/te_base
```

Set java property TE_BASE pointing to te_base folder:

```
vim ${tomcat}/bin/setenv.sh

Add following content to the file:
export JAVA_OPTS='-DTE_BASE=/home/USER/te_base'
```

Now, start the tomcat.

The TEAM Engine installation can be accessed via http://HOST:PORT/teamengine

4.1.3. Installation of WFS 2.0 test suite

Stop the tomcat.

Do a git clone of the ETS WFS 2.0 Source Code:

```
git clone https://github.com/opengeospatial/ets-wfs20.git
```

Checkout the desired version of ETS WFS 2.0. We recommend the newest release (currently 1.24):

```
git checkout ${project.version}
```

Build the source code with Maven:

```
clean install -Djava.awt.headless=true -DskipTests -Dmaven.compiler.target=1.7
-Dmaven.compiler.source=1.7
```

Go to 'target' folder:

```
cd target/
```

Unzip 'ets-wfs20-ctl.zip' to 'scripts' folder of 'te_base' directory:

```
unzip -o ets-wfs20-*-ctl.zip -d ~/te_base/scripts
```

Unzip 'ets-wfs20-deps.zip' to 'lib' folder of TEAM Engine webapp:

```
unzip -o ets-wfs20-*-deps.zip -d ${tomcat}/webapps/teamengine/WEB-INF/lib
```

Start the tomcat again.

The WFS 2.0 test suite is now listed on the main page of TEAM Engine (http://HOST:PORT/teamengine).

4.2. Test WFS 2.0 reference implementation with WFS 2.0 test suite

After having set up the WFS 2.0 reference implementation and the WFS 2.0 test suite, the reference implementation can be tested for compliance.

It is important that the TEAM Engine can reach the reference implementation via HTTP.

4.2.1. Execute a test of the reference implementation

1. Open http://HOST:PORT/teamengine with a web browser.
2. Click on button 'Create an account'.
3. Fill in a 'Username' and 'Password' and check the warning.
4. Click on 'Start Testing'.
5. Fill in your 'Username' and 'Password'.
6. Click on 'Create a new session'.
7. Select Organization 'OGC' and Specification 'WFS 2.0'.
8. Click on 'Start a new test session'.
9. Enter the capabilities URL of the tested RI (e.g. http://HOST:PORT/deegree/services/wfs?service=WFS&request=GetCapabilities).
10. Click on 'Start'.
11. Click on 'detailed test report'.
12. Inspect test results.

4.2.2. Examine the results of the test run

On the left hand side all test categories are displayed. They represent a conformance class of the WFS 2.0 specification, a related specification (e.g. filter conformance class of Filter Encoding specification) or any other groupable tests (e.g. Preconditions).

A category which is passed successfully is marked with green and a category with any test failures is marked with red.

You can click on any of the categories to see more details. If you do so, all test cases are displayed on the right hand side. For each test case the test name, a description including a reference to the relevant part of the specification, a start and duration time of the test is shown. You can click on 'Details' to view a description of the executed test.

In case a test is marked as failed (it is displayed in red) a reason for the test failure is printed. Click on the test name to see a more detailed description of the test execution. For example, the sent request and the received response are listed there.

With all those given information you should be able to interpret test failures. To really provide a reference implementation you should be able to fix all failures in the service implementation. This is sometimes located on the code side or the configuration of the software must be adjusted. Alternatively, you should be able to interpret and explain all existing failures and argue why your implementation does not break the specification.

Chapter 5. Suggestions for change requests deriving from realization of reference implementation

During the work on the WFS 2.0 reference implementation following uncertainties and shortcomings were discovered in the WFS 2.0 and FES 2.0 specification.

These findings are documented in following table and could be the basis for change requests.

Table 2. Uncertainties and shortcomings discovered in WFS 2.0 and FES 2.0 specification (suggestions for change requests)

Affected specification	Change
WFS 2.0.2	It is not possible to specify which operators (listed under ComparisonOperator, SpatialOperator and TemporalOperator in capabilities; for example Equals, Within, During) can be used for filters and/or joins functionalities in capabilities document.
WFS 2.0.2	It is not possible to specify which unit codes (e.g. RI or UCUM) can be used for units of measures (UOM) in capabilities document.
FES 2.0.2	The version action tokens described in the specification text (document 09-026r2, p. 33) list LATEST as a possible value. In the corresponding schema (http://schemas.opengis.net/filter/2.0/filter.xsd) LATEST is not included as possible value for VersionActionTokens but LAST is listed instead. The LATEST string in the specification text should be replaced with LAST.
WFS 2.0.2	In 15.3.5 (UpdateResults element) and 15.3.6 (ReplaceResults element) the attribute oldRid from FES 2.0.2 is referenced. The attribute is named previousRid in FES 2.0.2 and the corresponding schema (http://schemas.opengis.net/filter/2.0/filter.xsd). The string oldRid should be replaced with previousRid.

Appendix A: Revision history

Version	Author	Date	Comments
0.1	D. Stenger	2016-04	Created IER.
0.2	D. Stenger	2016-06	Created chapter structure and filled each section with first content.
0.3	L. Goltz	2016-08	Added installation details and description of some deegree configuration options required for WFS RI.
0.4	D. Stenger	2016-09	Filled all chapters with final content.
0.5	L. Goltz	2016-09	Made minor improvements regarding chapter content.
0.6	L. Bermudez	2016-10	Luis B review, typos and suggestions for other sections.
0.7	D. Stenger	2016-10	Added suggestions of reviews to ER and added new chapters.
1.0	D. Stenger	2016-10	Improved layout, updated metadata of document and filled index chapter with content. Completed DER.
OGC 16-025r2	L. Bermudez	2017-03	Moved history to appendix, updated note and title, cleaned abstract, created prefix.adoc to separate from main content, changed footnotes for hyperlinks, added bullet lists when missing, and other minor edits